Buletinul Ştiinţific al Universităţii Politehnica Timişoara

TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Volume 60(74), Issue 1, 2015

Analysis of Non-Preemptive Scheduling Techniques for HRT Systems

Mihai V. Micea[1] Cristina. S. Stangaciu[1] Vladimir I. Cretu[1]

Abstract – Special cases of hard-real time (HRT) scheduling mechanisms, which provide high predictability regarding task scheduling and execution, are studied in this paper. These mechanisms are all based on a proposed task model called *ModX*. Extensive evaluation tests have been performed to simulate and analyze the proposed scheduling algorithms and their comparative performance, which is also discussed in this paper.

Keywords: Scheduling, Embedded, Hard Real-Time, Non-Preemptive.

I. INTRODUCTION

Digital control is a topic of major interest in today's engineering and research activities. Embedded systems and digital signal processing (DSP) systems [1]-[4] are widely used in digital control applications, requiring, in most cases, real-time behavior of the hardware-software components. Many applications have a critical impact on the environment and/or on humans. Examples of such applications include: modern flight control systems, fly-by-wire, autopilot, automotive control, industrial mechatronics, nuclear plant surveillance, and so on.

There are two essential characteristics a hardware-software platform has to meet to provide correct operation results for critical applications [5]: (a) the entire process of system development should integrate the time coordinate, and (b) the system must provide maximum of predictability for the hard real-time tasks. As a key component of real-time application development and operation, task scheduling is closely related to the previously stated requirements.

Although a very large number and variety of scheduling techniques have been developed in the late years for both single processor and multiprocessor systems [6], hard real-time task scheduling with maximum of predictability still remains an open problem for critical applications. Some of the main reasons include the architectures which optimize the average case system operation (cache, pipelines, etc.), and the unrestricted use of interrupts and of the associated asynchronous mechanisms and tasks [7].

Our research focuses on developing suitable methodologies and architectures that enable hard real-time systems to meet the two basic requirements stated here. The approach is based on studying and integrating proper models of time, signals and tasks, emphasizing on non-preemptive scheduling techniques.

The next section introduces the model of hard real-time tasks, the *ModX*, based on which a number of non-preemptive scheduling techniques will be studied in Section III. The main results of the evaluation tests performed to simulate and analyze the proposed scheduling algorithms are presented in Section IV. A discussion on the non-preemptive scheduling techniques and their performance, current work and some prospects conclude the paper.

II. HARD REAL-TIME TASK MODEL

In a general acceptance, real-time applications (even those with critical operating requirements) contain both types of tasks – soft real-time (SRT) and hard real-time (HRT) tasks. Therefore, the development, scheduling and concurrent execution of the two types of tasks must be accommodated properly. In our approach, a task is classified as SRT if its correct operation is considered with respect to functional behavior only, while a HRT task also requires in addition a correct temporal behavior.

SRT tasks can therefore be modelled and analyzed using classical techniques; instead, the model of the HRT tasks must be able to describe and manipulate their temporal parameters. Thus, it must be considered with extreme care.

A *ModX* (executable module) is defined [8] as a periodic, modular, HRT task, with complete and strict temporal specifications, scheduled and executed in non-preemptive context:

$$M_i \equiv \langle \boldsymbol{T}, \boldsymbol{P}, \boldsymbol{S}, \boldsymbol{F} \rangle \quad (1)$$

where: $P = \{P_{IN}, P_{OUT}, P_{GLB}\}$ is the set of input, output and global parameters of M_i, respectively; $S = \{S_{IN}, S_{OUT}\}$ is the set of input and output signals M_i interacts with; F is the task's instruction set (its functional specification); and:

$$\boldsymbol{T} = \left\{ T_{pr}^{M_i}, T_{ex}^{M_i}, T_{dl}^{M_i}, T_{dy}^{M_i}, N^{M_i} \right\} \quad (2)$$

[1] Faculty of Automation and Computer Engineering, Computer Engineering and Information Technology Dept. Bd. V. Parvan 2, 300223 Timisoara, Romania, e-mail mihai.micea@cs.upt.ro

represents the set of temporal parameters of M_i, in their respective order: period, execution time, deadline, delay of execution during each period, and execution count.

Information exchange between the application *ModXs* is performed through the input, output and global parameters which define the set *P* (see (1)). *ModXs* can process input signals or can generate output signals, which formally define the *S* set. In the case of input signals, their temporal parameters define the behavior of the corresponding *ModXs*. The input signals (including the asynchronous events) are processed with our *ModX* model by periodic polling.

III. NON-PREEMPTIVE SCHEDULING ALGORITHMS OF INDEPENDENT MODX SETS

This section discusses the non-preemptive scheduling algorithms of hard real-time tasks on single-processor systems. Several cases are treated, starting from simple to more complex and realistic ones.

The task set model consists of simple and independent *ModXs*, each having the initial invocation time at $t_0 = 0$. Thus, each *ModX* M_i in the set can be characterized, according to (2), by:

$$\boldsymbol{T} = \left\{ T_{pr}^{M_i}, T_{ex}^{M_i}, T_{pr}^{M_i}, 0, \infty \right\} \tag{3}$$

In other words, the deadline of M_i equals its period, the execution delay during each period is null and the execution count states a continuous execution for M_i. The execution of M_i is not conditioned by any control or data dependencies with any other *ModXs* in the set.

Lemma 1. Let *M* be a set of simple and independent *ModXs*, characterized as in (3), and T_{LCM} the time interval equal to the least common multiplier of the *ModX* periods in *M*:

$$T_{LCM} = \min\left\{ C \,\middle|\, T_{pr}^{M_i} / C, \forall M_i \in \boldsymbol{M} \right\} \tag{4}$$

where: x/y means x divides y. If a particular algorithm is able to schedule the set *M* within the T_{LCM} interval, then *M* is feasible with respect to this scheduling algorithm.

Proof. The set *M* is composed of simple and independent *ModXs*, with their initial invocations aligned at the t_0 time instance. Moreover, the invocation time of all the *ModXs* are also aligned at each moment which is a common multiple of the task periods. On the other hand, the scheduling algorithms must guarantee that each *ModX* executes only once during each of its periods and without missing any of the specified deadlines. As a result, a cyclic behavior of the scheduling can be established based on the T_{LCM} interval.

Lemma 1 reduces the offline schedulability analysis of a set M of *ModXs* to a time interval of finite length, T_{LCM}.

Two main dynamic non-preemptive scheduling algorithms, considered as most efficient in the literature [9],[10], have been adapted to our task model: *MLFNP* (Minimum Laxity First Non-Preemptive) and *EDFNP* (Earliest Deadline First Non-Preemptive). Both have a general algorithmic framework, in which the *ModX* set is first sorted in non-decreasing order by period (i.e., for any pair of tasks M_i and M_j, if $i < j$, then $T_{pr}^{M_i} \le T_{pr}^{M_j}$). At any scheduling moment *t*, a *ModX* is selected for execution if it has not been already scheduled during its current period and if a particular criterion is verified:

(a) *MLFNP* selects the *ModX* with the minimum laxity (i.e. the time interval remaining available for the correct scheduling of the *ModX*, starting from *t*), as defined by:

$$L_i(t) = T_{pr}^{M_i}\left(\left\lfloor \frac{t}{T_{pr}^{M_i}} \right\rfloor + 1 \right) - T_{ex}^{M_i} - t \tag{5}$$

(b) *EDFNP* selects the *ModX* with the earliest deadline with respect to the current time *t*.

After a particular *ModX*, M_j, has been scheduled at time *t*, the scheduling time is increased with the execution time of M_j, and the procedure is reiterated until *t* reaches T_{LCM}.

An important advantage of the non-preemptive task models and scheduling techniques is that the offline analysis of the system feasibility is very close to the actual operating conditions at run-time, thus increasing the system predictability. The offline schedulability analysis can be speeded up by applying some necessity and/or sufficiency conditions instead of employing the algorithm to verify the feasibility of a task set.

The *ModX* model imposes some particularities to the schedulability conditions. Consider *M* a set of *n* *ModXs*, sorted in non-decreasing order by period. If *M* has a feasible schedule, then:

$$\text{CN1)} \quad \sum_{i=1}^{n} \frac{T_{ex}^{M_i}}{T_{pr}^{M_i}} \le 1 \tag{6}$$

This necessary condition is the basic relation that characterizes the feasible task scheduling on a single processor system. It states that the cumulative processor utilization cannot exceed unity. The second necessary condition has been demonstrated in [11]:

$$\text{CN2)} \quad \forall i.1 < i \le n; \forall L.T_{pr}^{M_1} < L < T_{pr}^{M_i} :$$

$$L \ge T_{ex}^{M_i} + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{T_{pr}^{M_j}} \right\rfloor \cdot T_{ex}^{M_j} \tag{7}$$

The condition (7) basically states that the processor utilization of a task set over any time interval *L* should not exceed that interval. Nevertheless, there is a difference between the task model considered in [11] and our *ModX* set, which is a concrete task set, with initial invocation times aligned to $t_0 = 0$. Therefore, examples of *ModX* sets can be found to be schedulable without satisfying CN2):

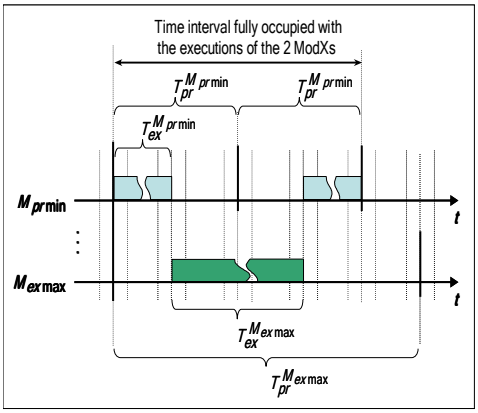

Fig. 1. Worst case for a feasible scheduling

$$\boldsymbol{M} = \left\{ M_i \equiv \left(T_{pr}^{M_i}, T_{ex}^{M_i} \right) \right\} = \\ = \{(10,4),(15,8),(90,4),(90,1)\} \tag{8}$$

For the *ModX* set in (8), which is schedulable with the *EDFNP* algorithm, the CN2) condition fails for $i = 2$ and $L = 11$.

Theorem 1. Let M be a set of n simple and independent *ModXs*, characterized as in (3). If M is schedulable, then:
CN3)

$$T_{ex}^{M_{ex\max}} \le 2\left(T_{pr}^{M_{pr\min}} - T_{ex}^{M_{pr\min}} \right) \tag{9}$$

where: $T_{ex}^{M_{ex\max}}$ is the execution time of the *ModX* with the maximum execution time in the set; $T_{pr}^{M_{pr\min}}$ and $T_{ex}^{M_{pr\min}}$ are the period and execution time, respectively, of the *ModX* with the minimum period in the set.

Proof. The theorem specifies a limiting condition for the maximum execution time of any *ModX* in M, with respect to the minimum *ModX* period in the set, assuming the execution without preemption of the *ModXs*.

The worst case for the execution (scheduling) of a feasible set M, regarding the two *ModXs* implied by the theorem, is presented in the figure above. It can be noticed that the time interval available for scheduling the M_{exmax} *ModX* without missing its deadlines is limited by the period and execution time of M_{prmin}.

Theorem 1 states the necessary condition added by our particular model of hard real-time task set to the non-preemptive scheduling analysis.

IV. PERFORMANCE OF THE NON-PREEMPTIVE ALGORITHMS

The performance evaluation of the non-preemptive scheduling algorithms discussed in the previous section focuses on determining the following parameters:

- The results of the schedulability conditions applied to the scheduling algorithm under test;
- The results of the schedulability analysis performed on randomly generated *ModX* sets. The analysis consists on applying the scheduling algorithm over the T_{LCM} interval calculated for the *ModX* sets under test (according to Lemma 1 and (4));
- The elapsed time of the schedulability analysis for each set of *ModXs*, on a PC type of workstation. This parameter characterizes only the general behavior of a particular scheduling algorithm during the offline analysis and differs from the run-time behavior parameters of the online scheduler.

Each set of *ModXs* is randomly generated, based on some general configuration parameters: n, the total number of *ModXs* in the set; the time interval which contains each of the *ModX* periods; the type of distribution used by the randomization algorithm to generate the periods – uniform distribution and normal (Gaussian) distribution; the rational values interval containing the processor utilization for the *ModX* set, $U^M = PU$; and the upper limit for the T_{LCM} value.

A comparative evaluation of the *MLFNP* and *EDFNP* scheduling algorithms has been performed, using the 12 workstations of the DSPLabs laboratory at UPT Timisoara (http://dsplabs.upt.ro). More than 24000 tests have been accomplished to calculate the *schedulability ratio (SR)* for the two algorithms, as a function of the following additional parameters: the total number of *ModXs* in the sets, {9, 15, 20}; the processor utilization PU, bounded by the following intervals: [0.6, 0.7], [0.7, 0.8], [0.8, 0.9] and [0.9, 1.0]; the *ModX* periods are randomly generated using the uniform and the normal distributions, with the upper limit of 310 and the lower limit of 10. As a result, the *ModXs* tested have a maximum ratio of 1/310 between the execution time and the period.

Although the second schedulability condition, CN2), does not apply properly to our *ModX* model (see discussion in Section III), we have included it in the evaluation tests (denoted as "*Jeffay*").

Figure. 2 presents some of the main results of the evaluation tests. The results show clearly that the *EDFNP* algorithm behaves much better than the *MLFNP* (i.e. the former issues a higher schedulability ratio than the latter), for all the cases considered: any *ModX* set dimension, any processor utilization PU, and any type of distribution used to generate the temporal parameters of the *ModXs*. The success ratio of both algorithms decreases when the processor utilization of the *ModX* sets is increased. On the other hand, the behavior of the algorithms improves when the number of *ModXs* in each set is increased. The reason is that, while the processor utilization remains constant, increasing the number of *ModXs* in a set implies a lowering of the execution times of each *ModX*. Therefore, the non-preemptive scheduling will

have more chances of success with "many, but smaller tasks" (higher task granularity) than vice versa.

Regarding the "*Jeffay*" test, the results show that *EDFNP* succeeds in scheduling many *ModX* sets for which the CN2) condition does not hold. This observation confirms our discussion about CN2), in Section III. On the other hand, *MLFNP* shows that the "Jeffay" test can be used as a valid condition for this algorithm in all the cases considered in our tests.

As previously mentioned, an upper bound parameter has been specified for the T_{LCM} value, calculated for each generated set of *ModXs*. This limitation is imposed because for sets of 20 *ModXs* for example, T_{LCM} can easily reach a magnitude order of 1030 and even more, generating a two-fold problem for our offline schedulability analysis approach:

a) The necessity of operating with very large numbers, which cannot be natively represented on PC architectures. As a result, specialized large integer arithmetic libraries must be used;
b) The time needed to perform the offline schedulability analysis is proportional with the size of T_{LCM}.

Some scheduling times obtained for sets of 18 *ModXs* with the limit of 2,000,000,000 for T_{LCM}, are shown in Table 1. The processor utilization has been set as low as possible (i.e. in the [1.0, 2.0] interval) to maximize the analysis times for the tested sets. The values in the table can be considered in a comparative manner, showing that the *EDFNP* algorithm is quicker than *MLFNP*.

Table 1. Elapsed times for some offline schedulability analysis tests

T_{LCM} values	Scheduling times [seconds]	
	MLFNP	EDFNP
145,044,900	476	469
325,155,600	1,060	1,052
149,189,040	483	481
681,912,000	2,214	2,212
1,730,907,360	5,698	5,601
Average values		
1,000,000,000	3,275	3,237

V. CONCLUSIONS

Critical and hard real-time applications require high operation predictability of the target system. Non-preemptive task models and scheduling techniques have been proven as a valid solution to develop and implement such applications on embedded and DSP-based platforms.

The offline feasibility analysis is a necessary step which eliminates the NP-hard type time and system resource requirements of an online analysis. Although reduced to a limited temporal interval (T_{LCM}) by using the Lemma 1, the offline schedulability analysis can be, in many cases, prohibitively time- (resource-) consuming. A set of schedulability conditions (necessary and/or sufficient conditions) can speed up the feasibility decision of some particular non-preemptive scheduling algorithm for a given task set.

Two of the most efficient dynamic non-preemptive scheduling algorithms have been adapted to our *ModX* model and studied: *MLFNP* and *EDFNP*. The performace evaluation tests have shown that *EDFNP* behaves better than *MLFNP*. Therefore, *EDFNP* has been chosen as the core of the online scheduling algorithms further developed to accommodate the realistic implementation of non-preemptive scheduling on real-time platforms.

The theoretical studies and test results showed that the CN2) schedulability condition, demonstrated in [11], does not apply to our *ModX* set model, which is a particular case of the task set considered in [11].

The non-preemptive task model and scheduling techniques presented in this paper are successfully being used in the development and implementation of a hard real-time kernel on a Motorola DSP56307 EVM platform [12][13]: the HARETICK kernel [5][14].

ACKNOWLEDGEMENTS

This work was partially supported by the strategic grant POSDRU/159/1.5/S/137070 (2014) of the Ministry of National Education, Romania, co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007-2013.

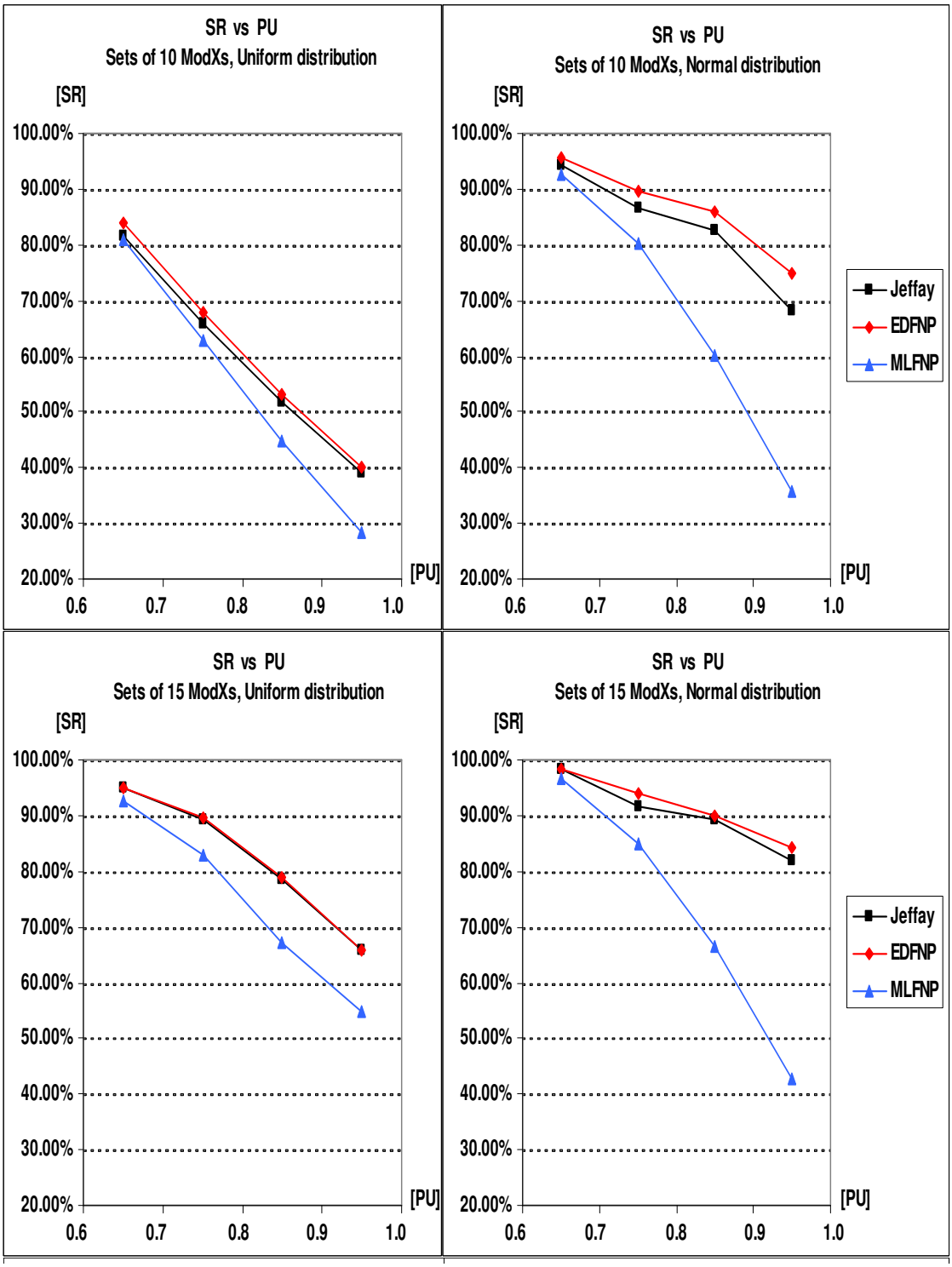

Fig. 2. *SR as a function of PU for the MLFNP and EDFNP algorithms*

REFERENCES

[1]. Lin, Chi-Ying, Li, Chien-Yao: *Design and Implementation of Advanced Digital Controls for Piezo-Actuated Systems using Embedded Control Platform*. In: Appl. Math 9.1L (2015), p. 251-258. [2] R. E. Collin, *Foundations for Microwave Engineering*, Second Edition, McGraw-Hill, Inc., 1992.

[2]. Antao, R., Mota, A., et al: *Adaptive control of a buck converter with an ARM Cortex-M4*. In: Proceedings of the 16th IEEE International Power Electronics and Motion Control Conference and Exposition, Antalya, 2014, p. 359

[3]. Morkoc, C., Onal, Y., et al: *DSP based embedded code generation for PMSM using sliding mode controller*. In: Proceedings of the 16th IEEE International Power Electronics and Motion Control Conference and Exposition, Antalya, 2014, p. 472

[4]. Puiu, D., Moldoveanu F. The Time Delay Ccontrol of a CAN Network with Message Recognition.In Bulletin of the Transilvania University of Braşov, Vol 3 (2010): 52, p.285-292.

[5]. Micea, M.V., Cretu, V.: *Non-Preemptive Execution Support for Critical and Hard Real-Time Applications on Embedded Platforms*. In: Proceedings of the International. Symposium on Signals, Systems and Electronics, Linz, 2004

[6]. Baruah, S., Bertogna, M., et al: *Multiprocessor Scheduling for Real-Time Systems*. Springer, 2015

[7]. Stewart, D. B.: *Twenty-five Most Common Mistakes with Realtime Software Development*. In Embedded Systems Conference, San Francisco, 2001.

[8]. Micea, M. V., Cretu, V., et al: *Program Modeling and Analysis of Real-Time and Embedded Applications*. In: Scientific Bulletin of "Politehnica" University of Timisoara, Transactions on Automatic Control and Computer Science. 49 (2004) No. 3, p. 207-212.

[9]. George, L., Rivierre, N., at al: *Preemptive and Non-Preemptive Real-Time Uni-Processor Scheduling*. In: *Rapport de recherche*, Nr. 2966, Institut National de Recherche en Informatique et en Automatique, INRIA, Rocquencourt, France, 1996.

[10]. Kang, S.I., Lee, H.K.: *Analysis and Solution of Non-Preemptive Policies for Scheduling Readers and Writers*. In ACM Operating Systems Review 32 (1998), p. 30-50.

[11]. Jeffay, K., Stanat, D., et al: *On Non-Preemptive Scheduling of Periodic and Sporadic Tasks*. In: Proceedings of the 12th IEEE Real-Time Systems. Symposium, San Antonio, p. 129.

[12]. Motorola, Inc.: *DSP56307: 24-Bit Digital Signal Processor: User's Manual*, DSP56307UM/D, Rev. 0, 08/10/98, Semiconductor Products Sector, DSP Division, Austin, USA, 1998.

[13]. Motorola, Inc.: *DSP56300: 24-Bit Digital Signal Processor: Family Manual*. Rev. 3, DSP56300FM/AD, Semiconductor Products Sector, DSP Division, Austin, USA, 2000.

[14]. Micea, M.V.: *HARETICK: A Real-Time Compact Kernel for Critical Applications on Embedded Platforms.* In: Proceedings of the 7th International Conference on Development and Application Systems, Suceava, 2004, p. 16.